Intrusion Detection System Visualization of Network Alerts

Dolores M. Zage and Wayne M. Zage
Ball State University

Final Report
July 2010
Sponsor: U.S. Army Research Laboratory

DEFENSE TECHNICAL INFORMATION CENTER

Information for the Defense Community

DTIC® has determined on 12/1/2016 that this Technical Document has the Distribution Statement checked below. The current distribution for this document can be found in the DTIC® Technical Report Database.

☒ **DISTRIBUTION STATEMENT A.** Approved for public release; distribution is unlimited. per ARO

☐ **© COPYRIGHTED**; U.S. Government or Federal Rights License. All other rights and uses except those permitted by copyright law are reserved by the copyright owner.

☐ **DISTRIBUTION STATEMENT B.** Distribution authorized to U.S. Government agencies only (fill in reason) (date of determination). Other requests for this document shall be referred to (insert controlling DoD office)

☐ **DISTRIBUTION STATEMENT C.** Distribution authorized to U.S. Government Agencies and their contractors (fill in reason) (date of determination). Other requests for this document shall be referred to (insert controlling DoD office)

☐ **DISTRIBUTION STATEMENT D.** Distribution authorized to the Department of Defense and U.S. DoD contractors only (fill in reason) (date of determination). Other requests shall be referred to (insert controlling DoD office).

☐ **DISTRIBUTION STATEMENT E.** Distribution authorized to DoD Components only (fill in reason) (date of determination). Other requests shall be referred to (insert controlling DoD office).

☐ **DISTRIBUTION STATEMENT F.** Further dissemination only as directed by (inserting controlling DoD office) (date of determination) or higher DoD authority.

Distribution Statement F is also used when a document does not contain a distribution statement and no distribution statement can be determined.

☐ **DISTRIBUTION STATEMENT X.** Distribution authorized to U.S. Government Agencies and private individuals or enterprises eligible to obtain export-controlled technical data in accordance with DoDD 5230.25; (date of determination). DoD Controlling Office is (insert controlling DoD office).

Table of Contents

List of Illustrations

Statement of the Problem: Long Term Goal

Develop a graphical representation of network alerts and events on a visual display that provides network intrusion detection analysts alternate views from the traditional displays.

Background

The United States Department of Defense continues to depend more and more on network based resources for information processing and data storage while network based attacks continue to increase. The size and complexity of networks are continuously increasing and security analysts face mounting challenges to secure and monitor their infrastructure for attacks [Goodall et al., 2005]. The number of network events and alerts analysts need to evaluate are increasing at an exponential rate. "This task is generally aided by an Intrusion Detection System (IDS), which attempts to automatically identify successful and unsuccessful attacks or abuse of computer systems" [Goodall et al., 2005 & McHugh, 2001]. As useful as an automated IDS is, they remain only a starting point. Security analysts must still use supplemental data sources to determine the accuracy and severity of an alert [Goodall et al., 2005]. Commonly, this entails the collection and identification of the "relevant details of network traffic related to the event being investigated" [Goodall et al., 2005]. The traditional process of viewing and evaluating alerts as page after page of text and numbers can be improved upon. Using a visual representation of network alerts may bring to light anomalies and intrusions that go overlooked while viewing network alerts in a traditional data view. False positives and unimportant network data may also be easily filtered out by the eye when viewing alerts on a visual display.

While there is a wealth of literature and research being conducted in the areas of visualization and information security, there are far fewer projects that combine the two. The field of security visualization is still in its relative infancy. However, even with this limited amount of information, there are a number of commonalities that emerge. For the purposes of this project, visualization can be thought of as the graphical representation of security log files. The majority of security tools output textual data and can generate millions of log entries daily. Research has suggested that it is difficult for the human brain to process large amounts of text rapidly and efficiently. Visualization offers a more effective approach to analyzing these log files by identifying outliers, detecting malicious activity, uncovering misconfigurations and anomalies, and spotting general trends and relationships among individual data points [Marty, 2009]. Visualization leverages the innate human capability for pattern recognition to increase the effectiveness, understanding, and speed of response to network security threats. "Visualization is a powerful link between the two most dominant information-processing systems, the human mind and the modern computer" [Nyarko, et al., 2002].

A visual approach offers a number of benefits over the traditional textual analysis of security data. Complex relationships can be hidden within the large amount of data produced by security tools. Instead of attempting to remember the relationships between individual data entries, an image can be produced that conveys these relationships in a

direct and concise form. Visualization can also assist security personnel in deciding what to investigate. Often, patterns that were not anticipated are revealed when the data are graphed. Exploring inconsistencies such as outliers and unusual traffic patterns between machines can lead to diagnoses that may have been missed. The current security landscape also dictates that decisions must be made rapidly. Visualization aids in the distillation of large amounts of data into something meaningful quickly. More meaningful data provides support for better decisions [Marty, 2009].

The process of analyzing and understanding security data offers an opportunity to present an example of the information seeking mantra. The information seeking mantra was introduced by Ben Shneiderman in 1996 and defines the way to gain insight from data as such:

"Overview first, zoom and filter, then details on-demand."

One can see the mantra being executed in the task model of Intrusion Detection work. Intrusion Detection is composed of three main tasks: monitoring, analysis, and response. Monitoring is "typically focused on the surveillance of the output of an IDS, involving the need for situational awareness" [Goodall et al., 2005]. The analysis task "focuses on determining the accuracy and severity of a security event uncovered in the monitoring task" [Goodall et al., 2005]. Response is the analyst's reaction to a security event. To understand the security data, an analyst needs to know the overall nature of the data – an overview. Based on the overview, interesting areas will need to be explored, i.e., an analyst may need to zoom in on a specific part of a graph. At the same time, certain data may need to be excluded by filtering. Then, to completely understand this area of interest, the original underlying data may need to be examined [Marty, 2009]. Once interesting patterns have been discovered, users require access to the actual data values [Nyarko et al., 2002]. Drill-down techniques are often useful to obtain details about particular items – these are the details on demand [Marty, 2009 & Nyarko et al., 2002].

It is important to note that the literature and all of the network security personnel we interviewed agreed that the context of an IDS alert must be preserved or, at the very least, available, in a resulting visualization. Analysts rarely (if ever) make a decision based solely on the information available from the IDS alert alone. While the exact process may differ between analysts, they all agreed that in the analysis task of IDS work, they attempted to gain a more complete picture of an alert by "reconstructing the event's timeline, the root cause of the event, and any related outcomes" [Goodall et al., 2005].

Intrusion Detection Systems and their resulting alerts are only one type of network security application and information and, therefore, only one piece of the security puzzle. Other important data sources and their respective analytical tools include packet captures (Wireshark, tcpdump), traffic flows (Argus, NetFlow), and firewalls (see Figure 1). Interviews with security analysts and relevant literature often stressed that these applications must be used in conjunction with one another and, to effectively manage security, none of them should be excluded. Utilizing these multiple data sources, as seen in Figure 1, will provide optimal coverage.

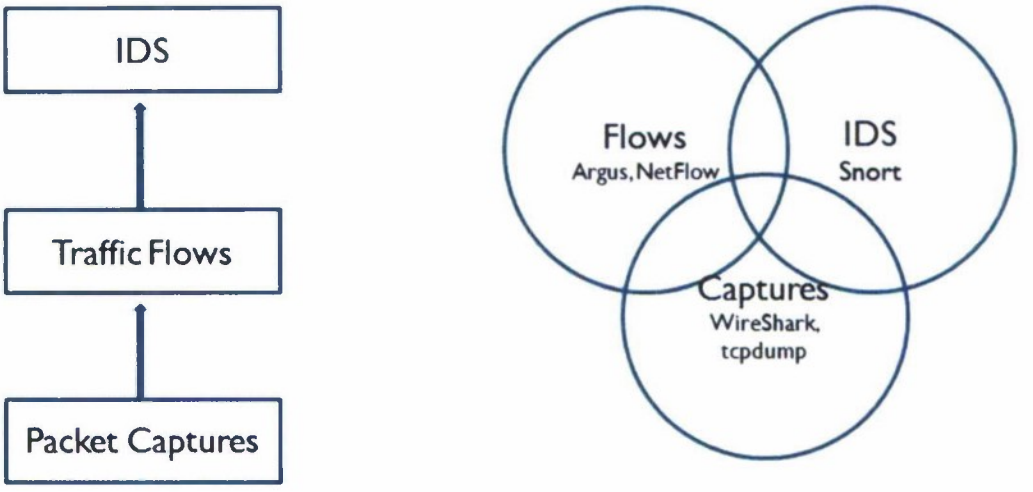

Figure 1. Network Security Data Sources. The relationship of the data sources and their levels of abstraction (left). Optimal coverage is achieved when all data sources are used (right).

Before discussing the types and specific examples of security visualization tools currently available, it may be worthwhile to discuss what has been referred to as the "dichotomy of security visualization" [Marty, 2005]. Thus far, security visualization tools have either been written by people that are network security personnel that do not know much about visualization theory and human computer interaction or by individuals with expertise in the field of visualization that do not know much about computer security. As a result, most current tools suffer from a lack of domain knowledge in one of these areas. An important goal for any future security visualization tool is to close this gap and create an application that is both user-friendly and easy to use and technically accurate and effective from a security perspective.

Another limitation of the tools that are currently in use is that they rarely handle multiple data sources and often require a separate application to parse the security data before it can be graphed. This is inconvenient for the user and makes evaluating security events in a timely fashion difficult. While these tools do work, "they are hard to use and consist of several steps using other numerous applications and consume a lot of time and even more computing power" [Swanson, 2008].

Because the field of security visualization is rapidly changing, providing an exhaustive list of all existing visualization solutions would be a daunting task. Our approach will be to provide an overview of the types of tools that are in use and list some representative examples. For simplicity, we will separate the tools into two categories: real-time and historical. Real-time applications plot whatever security data they visualize as the data occurs or is collected. Historical applications use previously collected data to produce a static graphical image with limited or no interactivity. All of the tools we cover are open source.

I. Real-time Security Visualization Tools

Snort Alert Monitor (SAM) utilizes a dashboard interface to provide real-time statistics unavailable with Snort alone. Its primary graph is a heatmap, as seen in Figure 2, which provides geographic locations for the IP addresses where the events that trigger Snort alerts originate from. It also provides a graph of alert severities and statistics on top attackers and high priority attacks.

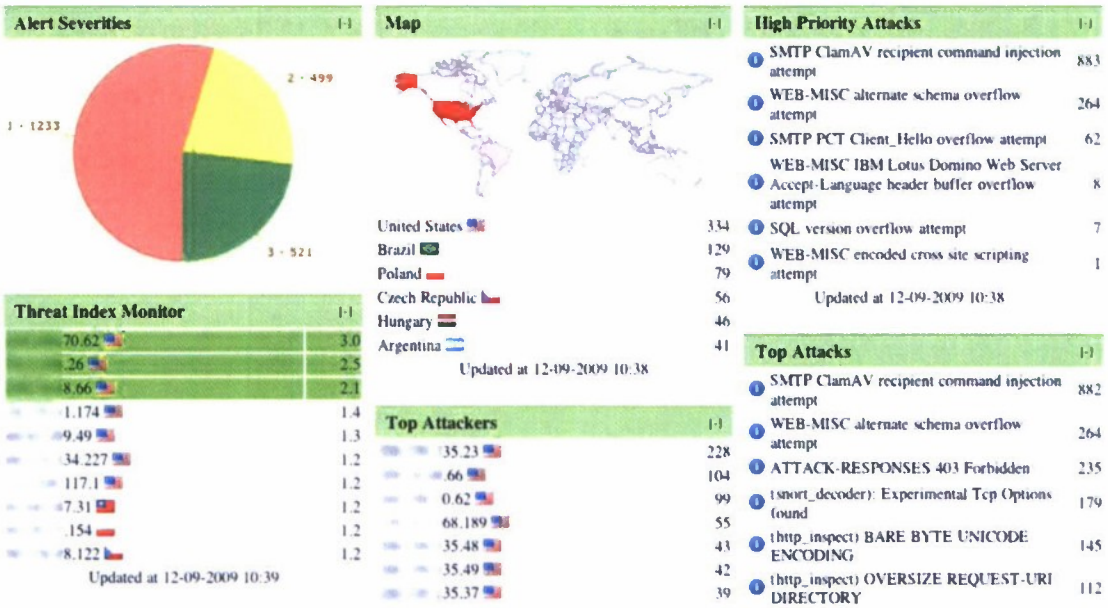

Figure 2. Snort Alert Monitor. An example of a dashboard interface.

Open Source Security Information Management (OSSIM) is another example of a dashboard interface. However, OSSIM bills itself as a Unified Threat Management tool and includes many open source security tools such as Intrusion Detection Systems, Vulnerability Assessment, and Anomaly Detection. It includes a number of different views, as seen in Figure 3, and provides multiple network statistics.

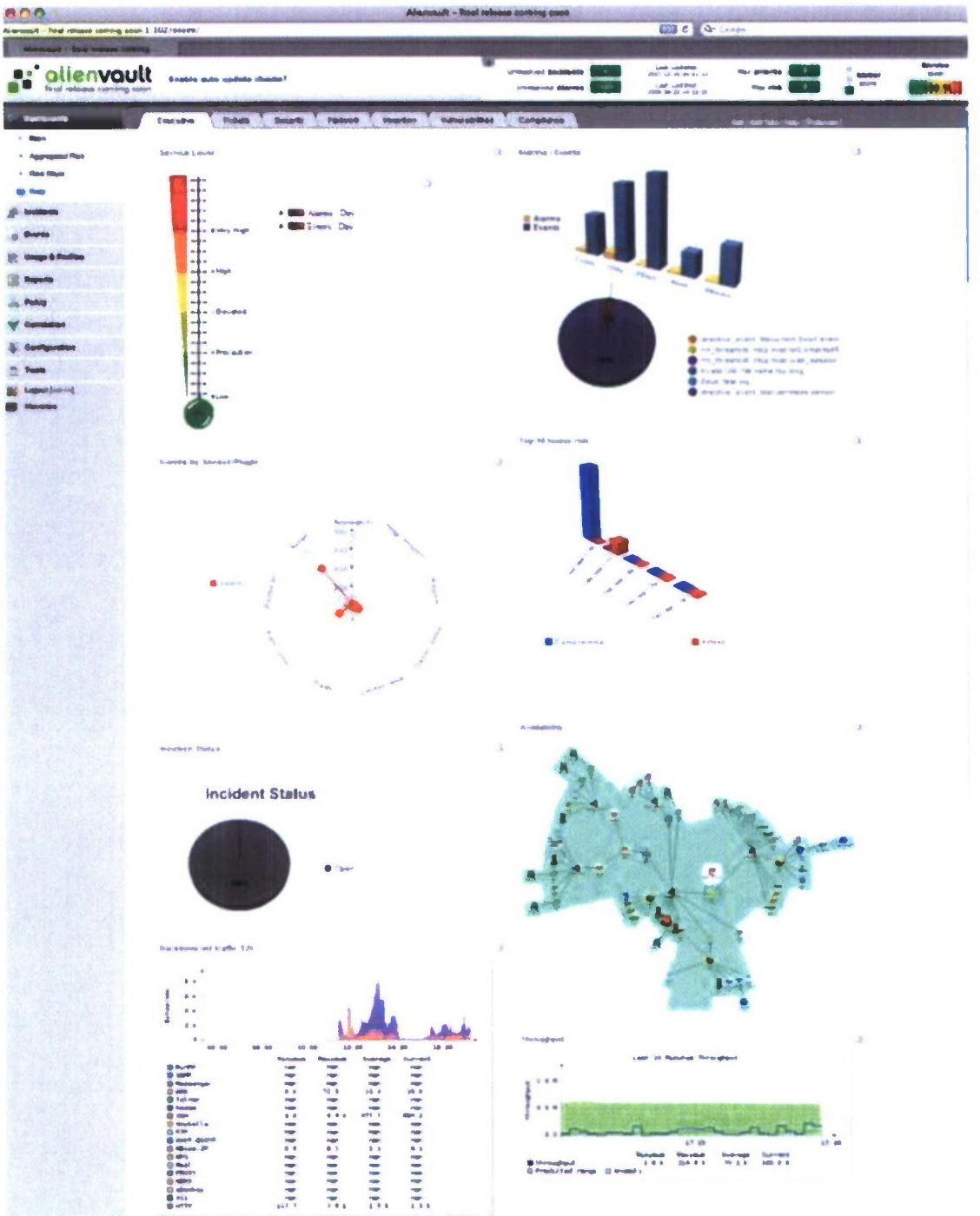

Figure 3. Open Source Security Information Management offers fast analysis and dashboard tools.

Etherape is an application that offers real-time visualization of packet captures. It can read traffic from a file as well as live from a network. Figure 4 displays Etherape output of network activity. It allows the user to differentiate protocols by color and filter the traffic to be shown. Hosts and links change in size with the amount of traffic.

Figure 4. Etherape provides real-time visualization of packet captures.

II. Historical Data Visualization Tools

Many of the historical data visualization tools are not intended specifically for security visualization. They can be used to graph data of other origin or nature, but because they are being used to graph security information and analysts find them useful, they will be listed here.

Afterglow is a command line application that generates link graphs. It expects comma-separated values (CSV) as input and will output DOT files, which can be used by other graphing applications, or can generate input for its own graphing library. Afterglow features node filtering by node name and frequency of occurrence, coloring of nodes and edges, sizing and assigning a shape to nodes, and aggregation of nodes. (http://afterglow.sourceforge.net/)

GraphViz is another tool that creates link graphs. It uses the DOT file format as input and implements a variety of layout algorithms for link graphs. (http://www.graphviz.org/)

R is a programming language and environment for statistical computing and graphics. It provides a wide variety of statistical and graphical techniques, and is highly extensible. It can read a variety of formats as input files, including CSV and SQL. It produces many types of graphs, including parallel coordinates, scatter plots, and treemaps. (http://www.r-project.org/)

GGobi is a general-purpose visualization application. It can use either CSV or XML files as input and can generate a number of graphs, but those most often used by security

personnel are parallel coordinates, link graphs, and scatter plots. Of particular interest is GGobi's ability to display simultaneous, multiple views of the same data, allowing for the exploration of different aspects of a data set. (http://www.ggobi.org/)

Mondrian is a general purpose statistical data-visualization system. The graphs it can create include scatterplots, maps, barcharts, histograms, parallel coordinates and boxplots. It uses CSV files as input and handles categorical data, geographical data and large data particularly well. (http://rosuda.org/mondrian/)

A collection of security visualization tools, including many of those in our overview, are available on the DAVIX Live CD available at http://www.secviz.org/. DAVIX allows users to try the tools without installing them to a hard drive.

Intermediate Term Objectives

After discussions with ARL network security personnel, it was decided that alerts from the Snort IDS tool would receive primary focus for our first prototype. The current ARL IDS Interrogator deposits the alerts to text-based log files. These log files will be parsed to drive the visualization process. Similar procedures can be applied to other alert log files. A prototype of the visualization tool (the Army Research Laboratory Visual Intrusion Detection System, henceforth ARL VIDS) was created and presented to ARL personnel for feedback. This feedback was incorporated and used to refine the prototype.

Schedule of Major Steps

1. Review of currently existing security visualization techniques and tools (discussed under the "Background for Long Term Goal" section)
2. Develop initial prototype using input from Snort alerts (discussed in the "Prototype Development and Progress" section
3. Update prototype incorporating feedback from ARL and investigation of other sources (discussed in the "Prototype Development and Progress" section)
4. Execute prototype using alerts from an IDS
5. Iterate Steps 3 and 4

Summary of Results: Prototype Development and Progress

The original ARL VIDS prototype focused on real-time visualization graphing Snort alerts in both 2D and 3D and linking the textual content from the alert log file. The prototype modeled and created alerts on the fly (see Figure 5). At this point, the prototype was not connected to an actual Snort log file. The Source IP Address was plotted on the y-axis, the Destination Port on the x-axis and for the 3D visualization time was plotted along the z-axis (see Figure 6). The user could customize the range of values plotted on the x-axis and have multiple windows open simultaneously. The visualizations had the capability to zoom in on an area of interest in the graph and the details could be accessed on demand by double clicking on a specific alert (see Figure 7).

This prototype was demonstrated at ARL on June 18, 2010 and valuable feedback was obtained that guided the development of the second prototype. End-user security personnel evaluation was crucial in determining which features communicated security information most effectively. It was suggested that elements of the 2D and 3D graphs be combined into a single 3D visualization. 3D was preferred as it provided another dimension on which to graph highly dimensional data. While the visualization literature is conflicted on whether 2D or 3D is better, it does provide some support for this preference. "The belief is that 2D images using graph-base representations although very useful do not scale well and have mapping and layout problems" [Swanson, 2008]. Movable coordinates or axes that remained in sync with the graphed alerts were requested. It was thought that a grid might also aid the user in maintaining orientation. Instead of time as a z-axis, a predetermined set of ranges that were mapped to specific types of alerts was offered as an alternative. The option to start plotting alerts from a specific time, for example, three hours previous to the current time and then any currently occurring alerts, was discussed. Details on demand could be accomplished by rendering the information directly on an alert in the display or by double clicking and selecting an alert and displaying the details.

Figure 5. Generated output used in the initial prototype.

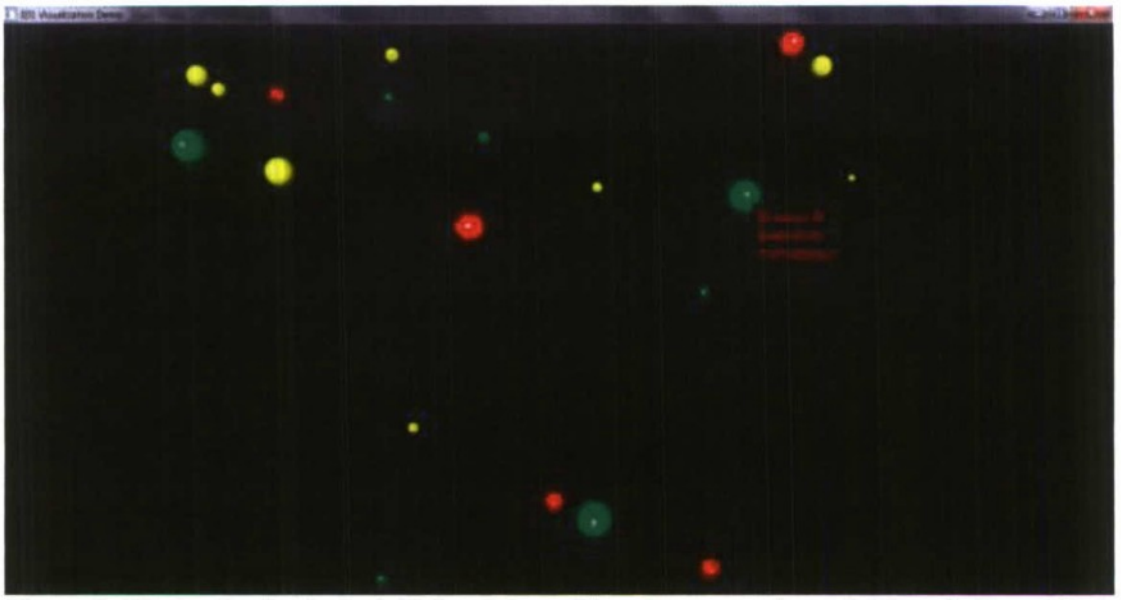

Figure 6. 3D visualization in the initial prototype. The spheres are color coded by priority and their size represents an aggregation of the number of alerts.

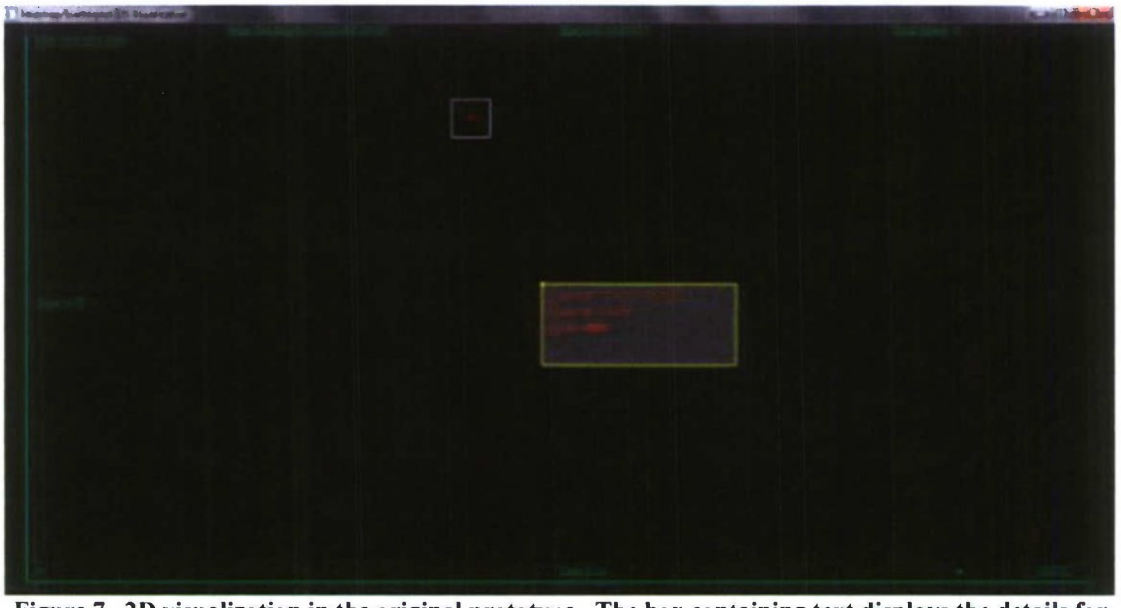

Figure 7. 2D visualization in the original prototype. The box containing text displays the details for the selected alert. The smaller box above highlights the most recent alert.

Our intentions were to develop a series of prototypes based on further research and feedback. An important feature of this study is to evaluate which visualizations are most effective, the Mackinlay criterion can prove useful. The first principle, the expressiveness criterion, states that "a set of facts is expressible in a visual language if the sentences (i.e., the visualizations) in the language express all the facts in the set of data, and only the facts in the data" [Marty, 2005]. A graph should encode data from the underlying data it represents. The second principle, the effectiveness criterion, states that a "visualization is more effective than another visualization if the information conveyed

by one visualization is more readily perceived than the information in the other visualization" [Marty, 2005].

In our scan of the current visualization tools, we explored the literature for the features requested by analysts. We also sought to interact with and gather information from end-user security personnel. The information provided by experienced security analysts and their evaluation of the prototype was crucial in determining which features communicated security information most effectively. In general, an effective VIDS should enable the user to:

- View and plot multiple categories and dimensions of the data collected
- Correlate historical data with current data to detect attacks carried out over an extended period of time
- View data in time slices, e.g. by the hour, day, month
- Customize the view to focus only on the data of interest: specific destination port ranges, destination subnets, etc.

In incorporating the feedback provided by ARL, careful attention was paid to creating a robust application that would create the most effective visualization possible, but also perform efficiently. Whereas the first prototype modeled and created its own pseudo alerts, the updated prototype connects to and parses an actual Snort log file in the alert_fast output format (see Figure 8). The process for connecting to and monitoring the log file emulates the unix tail function. After the initial reading of the log file, it is not read again unless it is modified. When it is read again, the file is read from the point at which it previously ended, subsequently, only new information is read. The application can detect if the log file has been truncated and will then perform a full read. These measures help optimize performance.

The 2D and 3D visualizations were combined and the z-axis values were created by parsing the Snort rules files which contain the signatures used to detect events and create alerts. Each individual signature was assigned a unique z-axis value and these were grouped into ranges by attack type (see Figure 9 for the mapping). As seen in the Settings dialog shown in Figure 12, the user could plot destination port, source port, destination IP address or source IP address on either the x-axis or y-axis. The range of values to graph on any of the axes could be adjusted by the user.

The user has the option of plotting alerts from a specific time and then continuing to plot new alerts as they occur. The display can be rotated, zoomed, and reset to allow the user multiple perspectives of the data (see Figure 11). By clicking an alert, the user can view its specific details (see figure 10). Once an alert is selected, the user can mark or tag it or choose to ignore it altogether. Transparency is used to keep alerts of a lesser depth from obscuring those with greater z values. The options that a user can select and change can be made local to each window, allowing multiple views that offer differing perspectives on the same data.

An attempt was made to incorporate a grid to provide the user with greater orientation, but it was thought to create confusion rather than greater clarity. It also violated a fundamental visualization principle: reducing non-data ink.

To enhance the broader participation of security analysts a video was created for a session using the ARL VIDS system. This video can be found online at:

http://rilittleford.iweb.bsu.edu/arlvids_hd.wmv

Analysts can view the short fifteen minute presentation and provide us more feedback on the demonstration.

A test installation of the Snort IDS has been deployed on a small local area network to begin collecting and creating alerts for the ARL VIDS tool to graph. Performance testing utilizing the linux tool netcat (and, alternatively, Snort's experimental "Unsock" alert-mode option) has been performed to send alert information across the network to be received and graphed by the ARL VIDS software. Snort's "-r" option has also been used to read publicly available, previously collected packet captures to generate real alerts for ARL VIDS to graph. The tcpreplay suite of tools is currently being used to replay packet capture files at arbitrary speeds onto the network for Snort to generate alerts from and ARL VIDS to then graph.

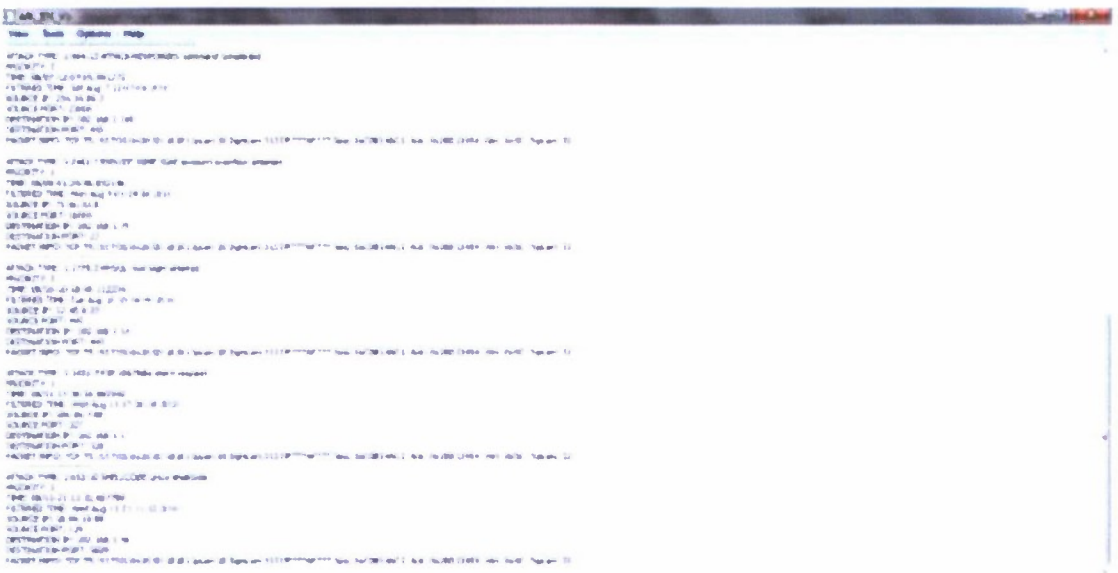

Figure 8. Alert information parsed in the current prototype.

Figure 9. Z-axis value ranges in the current prototype.

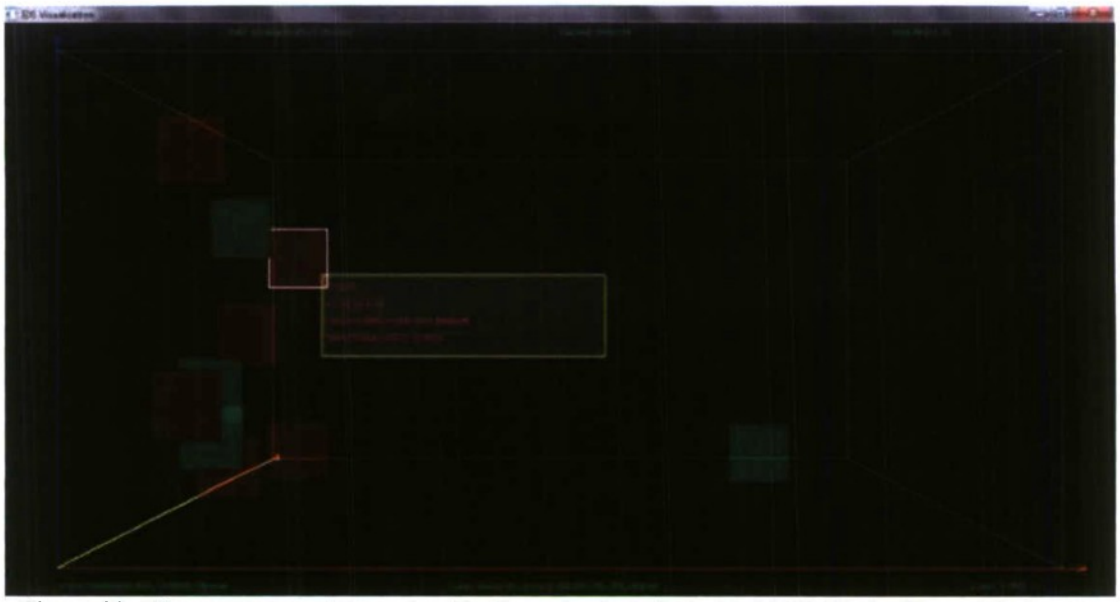

Figure 10. The current prototype visualization. A selected alert is highlighted with a white border and its details are displayed in text in the larger box.

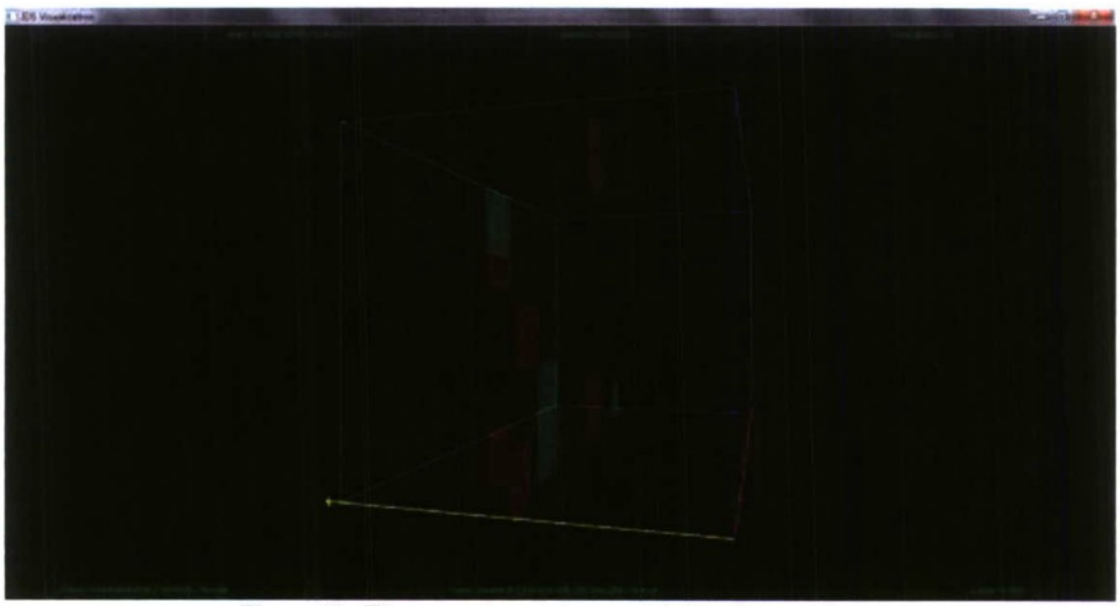

Figure 11. The current prototype rotated and zoomed out.

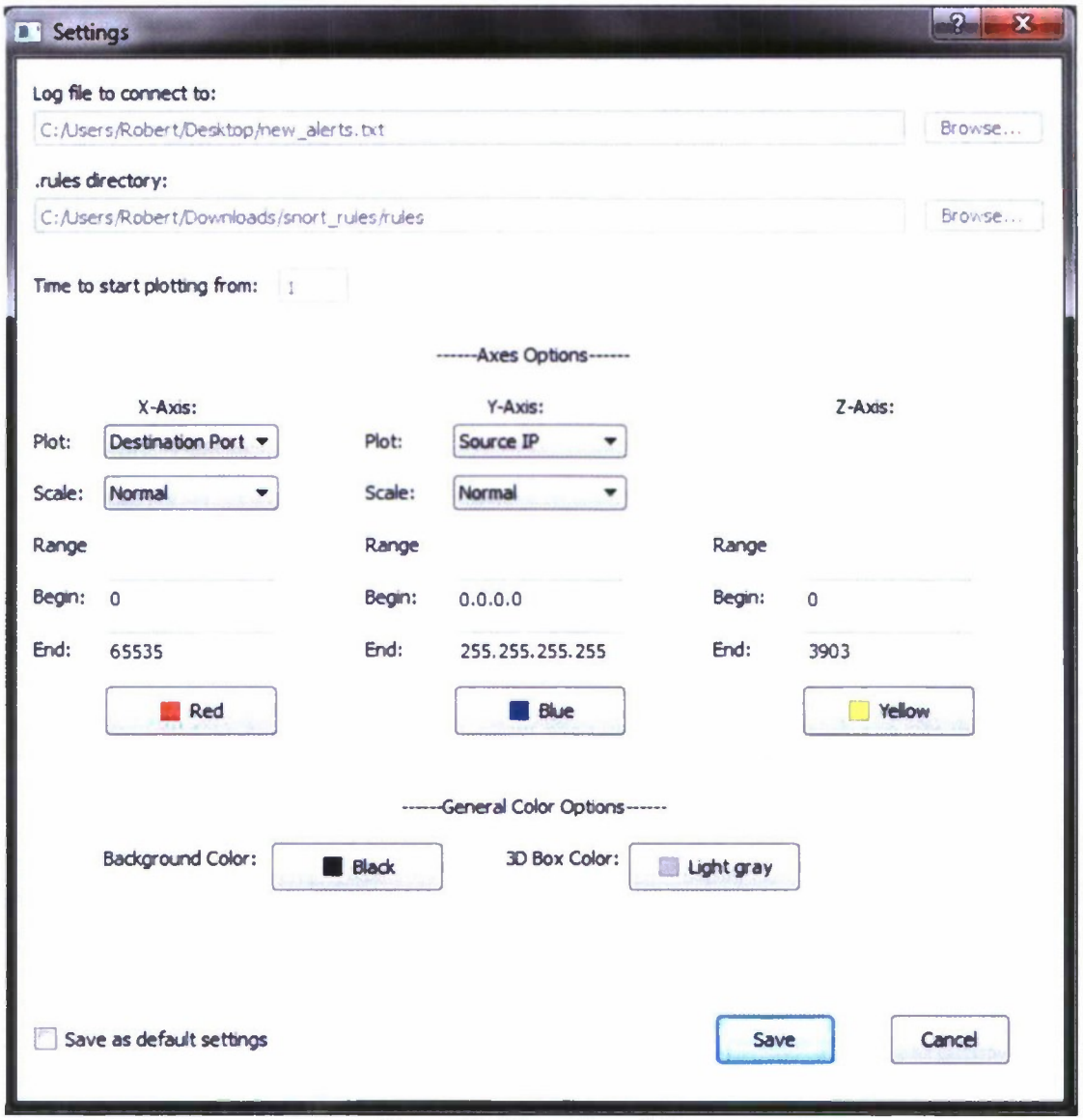

Figure 12. The settings dialog.

ARL VIDS can be expanded in the future to combine data from additional network security tools with that from Snort. Historical data can be correlated with the alerts occurring in real-time. Unique and rare event detection engines can be developed and utilized. Testing to determine how well ARL VIDS can visualize the alerts produced by a large network is also a high priority. Presently Ball State University (BSU) is the process of setting up a Snort installation for the BSU network. Our team has an excellent relationship with the university's security analysts and in the near future, our team will be given access to the alerts generated by this installation.

Dependencies

Ability to have access to the Interrogator detects database which includes access to various ARLCIMP customer network information.

Major Risks

While not classified, much of the ARL information being used is both proprietary and sensitive to the organizations that provide it. Mitigation approaches below are suggested:

1. Establish necessary proprietary information agreements and other contractual agreements with researcher(s) and their universities.
2. It is also suggested that the primary researcher and their team be either US citizens or have experience in conducting this type of analysis on DoD contracts.

Staff

Wayne Zage, Director of the S^2ERC and Professor, Department of Computer Science, Ball State University
Dolores Zage, Research Coordinator of the S^2ERC and Assistant Professor, Department of Computer Science, Ball State University
Blake Self, Research Assistant and former red team analyst, United States Marine Corp.
Robert Littleford, Research Assistant, Ball State University

Category of the Current Stage

Initial Concept Exploration

Contacts with Affiliates

Our primary contacts at ARL are Glenn Racine, Andrew J. Toth and Lee Trossbach. Lee Trossbach is credited with the initial concept for research and acts as consultant with Glen Racine on the direction and progress of the project. Two meetings at ARL were arranged and valuable guidance in requirements capture and evaluation of the draft user interface was obtained. Lee Trossbach also provided important information regarding typical Snort rules, network traffic and example packet captures.

Publications and other Research Products

Results of this research may have to be sanitized to not refer to any ongoing missions.

Future Research Directions

- Support for combining tools
- Unique analysis engine
- Rare event detection engine
- Heatmapping
- Signal Analysis (sound engine)
- Visualize data leaving network

References

Goodall, J. R., Lutters, W. G., Rheingans, P., and Komlodi, A. 2005. Preserving the Big Picture: Visual Network Traffic Analysis with TNV. In *Proceedings of the IEEE Workshops on Visualization For Computer Security* (October 26 - 26, 2005). VIZSEC. IEEE Computer Society, Washington, DC, 6.

Marty, R. 2008. *Applied Security Visualization*. 1. Addison-Wesley Professional.

McHugh, J. 2001. Intrusion and intrusion detection. In International Journal of Information Security (1): 14-35.

Nyarko, K., Capers, T., Scott, C., and Ladeji-Osias, K. 2002. Network Intrusion Visualization with NIVA, an Intrusion Detection Visual Analyzer with Haptic Integration. In *Proceedings of the 10th Symposium on Haptic interfaces For Virtual Environment and Teleoperator Systems* (March 24 - 25, 2002). HAPTICS. IEEE Computer Society, Washington, DC, 277.

Swanson, I. 2008. Malware, Viruses and Log Visualisation. In *Australian Digital Forensics Conference*. Paper 54.